ARL-TN-0865 ● JAN 2018

US Army Research Laboratory

Extraction of Vertical Profiles of Atmospheric Variables from Gridded Binary, Edition 2 (GRIB2) Model Output Files

by J L Cogan

ARL-TN-0865 ● JAN 2018

US Army Research Laboratory

Extraction of Vertical Profiles of Atmospheric Variables from Gridded Binary, Edition 2 (GRIB2) Model Output Files

by J L Cogan
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* January 2018	**2. REPORT TYPE** Technical Note	**3. DATES COVERED (From - To)** 30 October–30 November 2017
4. TITLE AND SUBTITLE Extraction of Vertical Profiles of Atmospheric Variables from Gridded Binary, Edition 2 (GRIB2) Model Output Files		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) J L Cogan		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-CIE 2800 Powder Mill Road Adelphi, MD 20783-1138		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TN-0865
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Vertical profiles of meteorological variables from model output data provide a means to ascertain the accuracy of the model compared to sounding data from observation systems such as radiosondes, radar profilers, unmanned aerial vehicles, and other atmospheric measurement systems. While various statistical packages provide a means to obtain upper air data from model output, many are fairly restrictive in terms of vertical extent and resolution, and some are large and complex. The method described here allows one to extract vertical profiles from Gridded Binary, edition 2 (GRIB2) model output at the inherent vertical resolution with relatively modest effort without recourse to complex software packages. It employs software that is readily available on many computer systems combined with relatively modest additional processing. Specifically, the method described herein uses wgrib2 commands along with a Python script or program to produce tabular text files that in turn may be processed using publicly available software on the US Army Research Laboratory GitHub site to generate “soundings” for user-defined levels and layers.

15. SUBJECT TERMS
vertical profile extraction, meteorological model comparison, model accuracy, user-defined vertical profiles, vertical profile analysis, Global Forecast System

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** James L Cogan
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UU	24	**19b. TELEPHONE NUMBER (Include area code)** (301) 394-2304

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Tables iv

Acknowledgments v

1. Introduction 1

2. Extraction and Processing of Model-Generated Vertical Profiles 1

    2.1 Create a Small Grid from the Input File 2

    2.2 Extract the Vertical Profile Data 2

    2.3 Convert the Output File into a User-Friendly Form 3

3. Convert Sounding to METCM or Other Layered Format 5

4. Summary and Conclusion 7

5. References 8

Appendix. Python process_wg2.py Code 9

List of Symbols, Abbreviations, and Acronyms 15

Distribution List 16

List of Tables

Table 1 Sample of output from “wgrib2 small_file.grb2 –v –s –lon 9.93 52.81 > ETGB_2017081506” (ETGB). Lines 61–70 are shown out of 417 lines. 3

Table 2 A sample of the output from the Python script applied to the file shown in Table 1. P is pressure, Hgt is height, Tmp is temperature, RH is relative humidity, and U and V are the horizontal components of the wind. 4

Table 3 Output with the same height structure as a METCM for the “sounding” of Table 2. Other layered forms or height structures may be generated by modifying the appropriate parameter file (e.g., usrmsg_lvls). In the table, the listed heights are the upper boundaries of the layers (zones or lines) except for line 0, which has values for the surface. The value -999 indicates missing data. Virt temp is virtual temperature and elevation is in meters. 6

Acknowledgments

I would like to acknowledge Brian Reen for his initial Python script and assistance with respect to the Python coding.

INTENTIONALLY LEFT BLANK.

1. Introduction

Vertical profiles of meteorological variables from model output data provide a means to ascertain the accuracy of the output compared to sounding data from observation systems such as radiosondes, radar profilers, unmanned aerial vehicles, and other atmospheric measurement systems. While various statistical packages provide a means to obtain upper air data from model output, many are fairly restrictive in terms of vertical extent and resolution and generally are large and complex. Examples include tools to process data from the National Oceanic and Atmospheric Administration (NOAA) National Operational Model Archive and Distribution System (NOMADS), which are available from NOAA sites such as https://www.ncdc.noaa.gov/nomads/tools-services. Other evaluation software tools are available at the Weather Research and Forecasting (WRF) Developmental Test Center (DTC) at websites such as https://dtcenter.org/met/users/ and https://dtcenter.org/upp/users/.

The method described here allows one to extract vertical profiles from Gridded Binary, edition 2 (General Regularly-distributed Information in Binary form or GRIB2) model output from the Global Forecast System (GFS) at the inherent vertical resolution with relatively modest effort (for information on GRIB2 see https://rda.ucar.edu/docs/formats/grib2/grib2doc/). It makes use of software that is readily available and can be implemented on many computer systems combined with relatively modest additional processing. Specifically, the method described herein uses standard wgrib2 commands (for an overall description, see http://www.cpc.ncep.noaa.gov/products/wesley/wgrib2/) along with a Python script or program to produce text files in an easy-to-understand tabular format. These output files may be processed using publicly available software (ARL_MET-profile_Converter) on the US Army Research Laboratory (ARL) GitHub site (https://github.com/usarmyresearchlab) to generate “soundings” for user-defined levels and layers.

2. Extraction and Processing of Model-Generated Vertical Profiles

This section describes the procedure to extract vertical “soundings” from GRIB2 model output files and process them into ASCII text files. Also, it very briefly outlines the process to generate vertical profiles of meteorological variables at user-defined levels and layers. To date, a few dozen GFS output files have been processed using wgrib2.

2.1 Create a Small Grid from the Input File

The first step is to create a very small GRIB2 (.grb2) file from the larger global or regional GRIB2 file. As used here, wgrib2 with –new_grid will generate a smaller grid interpolated from the fields of the parent grid (http://www.cpc.ncep.noaa.gov/ products/wesley/wgrib2/new_grid.html). The horizontal grid points of the smaller (new) grid are interpolated using bilinear interpolation of the larger grid data, unless otherwise specified (e.g., change to nearest neighbor). The command line to create the smaller file is as follows with uppercase denoting generic names such as for input or output files (e.g., OUTPUT_FILE).

Thus,

```
wgrib2 INPUT_GRIB2 –set_grib_type same –new_grid_winds earth –
new_grid latlon LON:X DIRECTION POINTS:DX(LON) LAT:Y
DIRECTION POINTS:DY(LAT) SMALLER_GRIB2,
```

where INPUT_GRIB2 is the input GRIB2 file, SMALLER_GRIB2 is the smaller output GRIB2 file, LON and LAT are the user-entered longitude and latitude in decimal degrees, X and Y DIRECTION POINTS refer to the number of grid points in the x- and y-directions, and DX and DY refer to the distance between grid points in the respective directions in units of longitude and latitude.

In the command line, “same” for –set-grib_type results in another grb2 file and “earth” for –new_grid_winds leads to winds relative to the Earth versus to the grid or undefined. “latlon” for –new_grid results in a new grid interpolated from the parent (old) grid, where the listed latitude and longitude are those for the new grid’s lower-left corner. Information on these and many other arguments used for wgrib2 may be found via http://www.cpc.ncep.noaa.gov/products/wesley/wgrib2/ long_cmd_list.html. An example for a location near the US East Coast is

```
wgrib2 gfs_4_20170404_000_012.grb2 –set_grib_type same –
new_grid_winds earth –new_grid latlon -76.12:2:0.0001 39.10:2:0.0001
small_file.grb2
```

In this example, the file small_file.grb2 has a 2-by-2 horizontal grid 0.0001° apart, which translates to about a 10-m separation.

2.2 Extract the Vertical Profile Data

The second step is to extract a sounding from the small grb2 file. That process is accomplished using another wgrib2 procedure:

```
wgrib2 SMALLER_GRIB2 –v –s –lon LON LAN,
```

where SMALLER_GRIB2 is the output from the procedure of Section 2.1, –v refers to verbose output (includes the data values), –s refers to simple inventory (listing of the output variables and parameters), and –lon produces data for the nearest grid point to the stated longitude and latitude. With a grid separation of approximately 10 m, the values are essentially at the stated coordinates. This wgrib2 process produces a list of the variables, data values, and other information for the output grid point and prints it on the screen. To save the output, redirect the data to a separate file.

Using the above example,

```
wgrib2 small_file.grb2 –v –s –lon -76.41 39.10 >SiteA_profile.
```

2.3 Convert the Output File into a User-Friendly Form

The output from the wgrib2 process of Section 2.2 is very wordy and not readily useable for additional processing, such as to create a vertical profile in a format similar to a computer meteorological message (METCM). Herein METCM refers to tabular output with the same height and layer (aka zone) structure as the standard METCM described in US Army FM 3-09.15 (2007). Table 1 shows a sample of output for Bergen, Germany (ETGB), from the second wgrib2 procedure. The nominal start time of the model was at 00 coordinated universal time (UTC) on 15 August 2017 and the data shown were from the 6-h forecast (i.e., for 0600 the same day).

Table 1 Sample of output from "wgrib2 small_file.grb2 –v –s –lon 9.93 52.81 > ETGB_2017081506" (ETGB). Lines 61–70 are shown out of 417 lines.

```
61:12600:d=2017081500:VGRD V-Component of Wind [m/s]:50 mb:6 hour
fcst::lon=9.930000,lat=52.810000,i=1,ix=1,iy=1,val=3.89432
62:12810:d=2017081500:ABSV Absolute Vorticity [1/s]:50 mb:6 hour
fcst::lon=9.930000,lat=52.810000,i=1,ix=1,iy=1,val=0.000109774
63:13020:d=2017081500:O3MR Ozone Mixing Ratio [kg/kg]:50 mb:6 hour
fcst::lon=9.930000,lat=52.810000,i=1,ix=1,iy=1,val=3.67529e-06
64:13230:d=2017081500:HGT Geopotential Height [gpm]:70 mb:6 hour
fcst::lon=9.930000,lat=52.810000,i=1,ix=1,iy=1,val=18817.7
65:13440:d=2017081500:TMP Temperature [K]:70 mb:6 hour
fcst::lon=9.930000,lat=52.810000,i=1,ix=1,iy=1,val=219.231
66:13650:d=2017081500:RH Relative Humidity [%]:70 mb:6 hour
fcst::lon=9.930000,lat=52.810000,i=1,ix=1,iy=1,val=4.97867
67:13860:d=2017081500:UGRD U-Component of Wind [m/s]:70 mb:6 hour
fcst::lon=9.930000,lat=52.810000,i=1,ix=1,iy=1,val=4.55109
68:14070:d=2017081500:VGRD V-Component of Wind [m/s]:70 mb:6 hour
fcst::lon=9.930000,lat=52.810000,i=1,ix=1,iy=1,val=6.59625
69:14280:d=2017081500:ABSV Absolute Vorticity [1/s]:70 mb:6 hour
fcst::lon=9.930000,lat=52.810000,i=1,ix=1,iy=1,val=0.000108671
70:14490:d=2017081500:O3MR Ozone Mixing Ratio [kg/kg]:70 mb:6 hour
fcst::lon=9.930000,lat=52.810000,i=1,ix=1,iy=1,val=2.05373e-06
```

A Python 3 program (often called a Python 3 script) was written to convert the output into a readily readable and useable form. It reads in the aforementioned output file from wgrib2 (ETGB_2017081506, in the example), extracts appropriate information, and lists the extracted information in a readable tabular form. The Python script used here is described in the Appendix.

To run the script, type

```
python3 process_wg2.py INPUT_FILE
```

For the input file that included the data in Table 1, use the following:

```
python3 process_wg2.py ETGB_2017081506
```

which produces the output file ETGB_2017081506_out.

GFS output files from NOAA starting on 12 UTC 11 May 2016 extend up to 1.0 hPa as in Table 2, but before then the last data line ended at 10 hPa. The aforementioned methods work for both.

Table 2 A sample of the output from the Python script applied to the file shown in Table 1. P is pressure, Hgt is height, Tmp is temperature, RH is relative humidity, and U and V are the horizontal components of the wind.

6-h forecast after model start at: 2017081500
Latitude: 52.810000 Longitude: 9.930000

P (hPa)	Hgt (m)	Tmp (K)	RH (%)	U (m/s)	V (m/s)
1009.8	61.5	291.68	70.2	–2.77	2.09
1000.0	145.2	291.43	65.6	–4.79	4.38
975.0	362.0	291.19	61.3	–4.21	8.61
950.0	585.0	292.04	58.2	0.37	9.11
925.0	813.9	291.23	61.8	2.36	7.05
900.0	1048.2	289.96	65.7	2.30	6.17
850.0	1532.9	286.24	78.0	1.28	7.47
800.0	2039.5	282.00	83.7	0.44	9.11
750.0	2570.1	277.64	77.0	0.38	9.06
700.0	3128.8	274.77	50.6	4.77	6.19
650.0	3723.7	273.13	15.7	7.82	3.89
600.0	4359.9	269.44	14.9	8.19	3.55
550.0	5040.7	264.97	18.3	8.70	4.83
500.0	5773.0	259.82	23.9	8.94	6.46
450.0	6564.8	253.35	58.9	9.53	8.05
400.0	7426.9	246.73	94.8	10.45	8.23
350.0	8378.0	239.70	98.6	11.04	7.78
300.0	9440.3	230.95	98.5	12.09	8.78
250.0	10645.5	220.83	99.8	13.01	9.66
225.6	11302.4	216.77	50.0	15.84	8.85
200.0	12068.1	218.63	26.5	17.69	11.15
165.4	13300.0	222.40	50.0	17.50	16.18
150.0	13933.3	221.10	1.9	16.13	15.94
100.0	16529.6	218.50	2.7	9.90	10.46
70.0	18817.7	219.23	5.0	4.55	6.60
50.0	20977.7	220.07	2.6	1.01	3.89

Table 2 A sample of the output from the Python script applied to the file shown in Table 1 (continued)

P (hPa)	Hgt (m)	Tmp (K)	RH (%)	U (m/s)	V (m/s)
30.0	24279.5	222.30	0.5	–2.38	1.77
20.0	26945.0	226.65	0.3	–4.19	1.70
10.0	31608.0	233.32	0.1	–6.48	–0.12
7.0	34064.4	237.30	0.0	–8.76	0.40
5.0	36423.1	241.92	0.0	–9.96	0.67
3.0	40106.3	250.72	0.0	–12.85	1.57
2.0	43117.8	256.42	0.0	–5.68	3.00
1.0	48361.6	259.42	0.0	–22.37	4.14

3. Convert Sounding to METCM or Other Layered Format

The ARL_MET-profile_Converter programs (https://github.com/usarmyresearch lab) or similar ones may be used to produce a table of user-defined levels or layers from the table of "sounding" data (e.g., Table 2). Cogan (2017a, 2017b) and included references that describe this C program and its application present samples of output. One version produces output for height levels and layers, and a second provides output for pressure levels and layers. The user provides a text file that contains the height or pressure levels, respectively, which also serve as the vertical boundaries of the included layers. The comparisons involving conversion of GFS-derived soundings into a METCM used only a version for height levels and layers, which is briefly discussed in the rest of this section.

The input and output directories are defined in the input_parameters file, which is in the same directory as the C program executable. For a METCM, the file metcm_lvls contains the boundary levels of the METCM layers (zones) starting with the surface through 30 km. It is also in the same directory as the C program. Note that zone 0 (the first data line) has values for the surface only. Consecutive zones have weighted mean layer values. Sensible temperature was included in addition to the standard METCM variables, as seen in Table 3. The program is run using the following command line:

```
./convertgfs INPUT_FILE
```

For the example in this report, the line would read

```
./convertgfs ETGB_2017081506_out
```

Note that some operating systems may not use the "./" before the executable name. Table 3 contains values of the listed variables for the height layers of a METCM for the sounding of Table 2 (ETGB). Wind direction is in tens of mils and 6400 mils = 360° = 640 tens of mils.

Table 3 Output with the same height structure as a METCM for the "sounding" of Table 2. Other layered forms or height structures may be generated by modifying the appropriate parameter file (e.g., usrmsg_lvls). In the table, the listed heights are the upper boundaries of the layers (zones or lines) except for line 0, which has values for the surface. The value -999 indicates missing data. Virt temp is virtual temperature and elevation is in meters.

METCM output

Date: 20150505 Time: 6 Latitude: 52.81000 Longitude: 9.93000
Elevation: 68.3 Ceiling: -999 Visibility: -999

Line	Height (m)	Wind direction (tens of mils)	Wind speed (kt)	Virt temp (K*10)	Pressure (mb)	Temperature (K*10)
0	0	297	17	2920	991	2902
1	200	305	23	2915	980	2898
2	500	329	33	2906	951	2890
3	1000	380	39	2912	908	2898
4	1500	398	33	2882	856	2868
5	2000	389	29	2839	806	2828
6	2500	391	31	2803	759	2793
7	3000	406	36	2771	713	2763
8	3500	413	43	2739	671	2733
9	4000	415	47	2707	630	2702
10	4500	415	49	2675	591	2671
11	5000	414	50	2643	554	2639
12	6000	412	49	2591	502	2588
13	7000	408	47	2520	440	2518
14	8000	408	47	2445	383	2444
15	9000	422	55	2368	332	2368
16	10000	437	73	2293	287	2293
17	11000	439	81	2224	247	2224
18	12000	435	67	2170	211	2170
19	13000	433	55	2156	180	2156
20	14000	431	44	2161	154	2161
21	15000	431	36	2162	131	2162
22	16000	432	29	2161	112	2161
23	17000	430	22	2158	96	2158
24	18000	421	18	2150	82	2150
25	19000	410	13	2142	70	2142
26	20000	413	8	2140	59	2140
27	22000	402	3	2141	47	2141
28	24000	273	5	2154	34	2154
29	26000	240	12	2180	25	2180
30	28000	225	18	2218	18	2218
31	30000	205	22	2265	13	2265

4. Summary and Conclusion

This brief report presents a method to extract vertical profiles of meteorological variables from GRIB2 GFS output files and convert them into a standard type of format. Further processing described in detail in Cogan (2017a, 2017b) and the included references may be used to convert these “soundings” into profiles of user-defined level and layer height or pressure values.

The wgrib2 program is available from NOAA and can be installed on many computers. The Python program was written on a Linux computer with Python 3.5. The C program produces profiles for user-defined height or pressure levels and layers, and earlier versions have been used in several model evaluations.

The method described here provides a means to prepare GFS output for comparison to data from observation systems or other models. An early attempt suggests that wgrib2 may be suitable for GRIB2 output from other models (e.g., the Global Air Land Weather Exploitation Model), but would require changes to the argument lists and the variable table, and/or need one or more additional wgrib2 functions.

5. References

Cogan J. Evaluation of model-generated vertical profiles of meteorological variables: method and initial results. Meteorol Appl. 2017a;24:219–229.

Cogan J. Model evaluation using ballistic trajectories and preliminary mesoscale model accuracies with age of global model initialization data. Meteorol Appl. 2017b. doi:10.1002/met.1684.

US Army FM 3-09.15/Marine Corps MCWP 3-16.5. Artillery meteorology. Washington (DC): Headquarters, Department of the Army; 2007.

Some relevant web (URL) sites include the following:

- ARL GitHub site: https://github.com/usarmyresearchlab.
- An overall description of wgrib2 may be found at http://www.cpc.ncep.noaa.gov/products/wesley/wgrib2/. Information on arguments used for wgrib2 may be found at http://www.cpc.ncep.noaa.gov/products/wesley/wgrib2/long_cmd_list.html.
- NOMADS tools and services may be obtained at https://www.ncdc.noaa.gov/nomads/tools-services. Additional evaluation software tools are available at the WRF DTC via https://dtcenter.org/met/users/, https://dtcenter.org/upp/users/ and their included links.

Appendix. Python process_wg2.py Code

This Appendix lists the Python process_wg2.py code as written for this application. It represents one approach to extracting and reformatting data using Python.

```
#!/ bin/env python3

import re
import sys
from collections import defaultdict

#NOTE: sys.argv[0] is the program (e.g., process_data.py).

with open(sys.argv[1], "r") as f:
    input_data = f.readlines()

output_file = sys.argv[1] + "_out"
print('Reading from file: ', sys.argv[1])
pa_to_hPa = 0.01  # 1 hPa = 1 mb

p_val = defaultdict(dict)
other_val = defaultdict(dict)
pressvals=set()

for currentline in input_data:

   match = re.search(':surface', currentline)
   if match:
      match = re.search("PRES", currentline)
      if match:
         currentline_list = re.split('[=]', currentline)
         lon = currentline_list[2].replace(',lat','')
         lat = currentline_list[3].replace(',i','')
         if(float(lon) > 180):
           lon = str(float(lon) - 360)
         current_surfline = currentline_list[1]
         currentline_surf = re.split('[:]', current_surfline)
         model_start = currentline_surf[0]             #Get the model start date and time.
         fcst_time = currentline_surf[3]               #Get time of forecast from model start.
         if(fcst_time == 'anl'):                       #If = anl (analysis) then 0-h forecast.
            fcst_time = '0 hour'
         else:
            fcst_time = fcst_time.replace('fcst', '')
         current_surface = currentline_surf[2].replace(':','')   #Remove : from surface value.
         current_surf_press = currentline_list[7]              #Get the surface pressure value.
         surface_pressure = float(current_surf_press)*pa_to_hPa  #Convert to float for hPa value.
         #print("surface_pressure ", surface_pressure)

      match = re.search(':HGT', currentline)
      if match:
          currentline_list = re.split('[=]', currentline)
          current_surf_hgt = currentline_list[7]
          other_val['HGT'][current_surface]=current_surf_hgt
    match = re.search(':2 m above ground:', currentline)
    if match:
      currentline_list = re.split('[=]', currentline)

      match = re.search(':TMP', currentline)
      if match:
         current_surf_temp = currentline_list[7]
         other_val['TMP'][current_surface]=current_surf_temp  # Actually 2 m AGL temperature.

      match = re.search(':RH', currentline)
      if match:
         current_surf_rh = currentline_list[7]
         other_val['RH'][current_surface]=current_surf_rh  # Actually 2 m AGL relative humidity.

    match = re.search(':10 m above ground:', currentline)
    if match:
      currentline_list = re.split('[=]', currentline)

      match = re.search(':UGRD', currentline)
      if match:
         current_surf_u = currentline_list[7]
         other_val['UGRD'][current_surface]=current_surf_u  # Actually 10m AGL U-component of
wind.

      match = re.search(':VGRD', currentline)
      if match:
         current_surf_v = currentline_list[7]
         other_val['VGRD'][current_surface]=current_surf_v  # Actually 10 m AGL V-component of
wind.

    match = re.search(':tropopause', currentline)
    if match:
      currentline_list = re.split('[=]', currentline)
      current_trop_line = currentline_list[1]
      currentline_trop = re.split('[:]', current_trop_line)
      current_trop = currentline_trop[2]
      match = re.search(':HGT', currentline)
      if match:
         current_trop_hgt = currentline_list[7]
         other_val['HGT'][current_trop]=current_trop_hgt

      match = re.search('PRES', currentline)
      if match:
         current_trop_prs = currentline_list[7]
         other_val['PRES'][current_trop]=current_trop_prs
         tropo_pressure = float(current_trop_prs)*pa_to_hPa  #Convert to float for hPa value.

      match = re.search('TMP', currentline)
      if match:
         current_trop_tmp = currentline_list[7]
         other_val['TMP'][current_trop]=current_trop_tmp
      match = re.search('UGRD', currentline)
      if match:
         current_trop_u = currentline_list[7]
         other_val['UGRD'][current_trop]=current_trop_u

      match = re.search('VGRD', currentline)
      if match:
         current_trop_v = currentline_list[7]
         other_val['VGRD'][current_trop]=current_trop_v
         other_val['RH'][current_trop] = 50      # No RH data line for tropopause. Set at some value.

   match = re.search(':max wind', currentline)
   if match:
      currentline_list = re.split('[=]', currentline)
      current_maxwind_line = currentline_list[1]
      currentline_maxwind = re.split('[:]', current_maxwind_line)
      current_maxwind = currentline_maxwind[2]

      match = re.search(':HGT', currentline)
      if match:
         current_maxwind_hgt = currentline_list[7]
         other_val['HGT'][current_maxwind]=current_maxwind_hgt

      match = re.search(':PRES', currentline)
      if match:
         current_maxwind_prs = currentline_list[7]
         other_val['PRES'][current_maxwind]=current_maxwind_prs
         maxwind_pressure = float(current_maxwind_prs)*pa_to_hPa  #Convert to float for hPa
value.

      match = re.search(':TMP', currentline)
      if match:
         current_maxwind_tmp = currentline_list[7]
         other_val['TMP'][current_maxwind]=current_maxwind_tmp

      match = re.search(':UGRD', currentline)
      if match:
         current_maxwind_u = currentline_list[7]
         other_val['UGRD'][current_maxwind]=current_maxwind_u

      match = re.search(':VGRD', currentline)
      if match:
         current_maxwind_v = currentline_list[7]
         other_val['VGRD'][current_maxwind]=current_maxwind_v
         other_val['RH'][current_maxwind] = 50      # No RH data line for tropopause. Set at some
value.

# Begin "regular" data lines arranged by pressure levels.

   match = re.search('mb:', currentline)
   if match:
      currentline_list = re.split('[=]', currentline)
      current_pline = currentline_list[1]
      currentline_prs = re.split('[:]', current_pline)
      current_press = currentline_prs[2].replace(' mb','') #Remove mb from pressure value.
      pressvals.add(float(current_press))

      match = re.search(':HGT', currentline)
      if match:
        currentline_hval = currentline_list[7]
        p_val['HGT'][str(int(current_press))]=currentline_hval

      match = re.search(':TMP', currentline)
      if match:
        currentline_tval = currentline_list[7]
        p_val['TMP'][str(int(current_press))]=currentline_tval

      match = re.search(':RH', currentline)
      if match:
        currentline_tval = currentline_list[7]
        p_val['RH'][str(int(current_press))]=currentline_tval

      match = re.search(':UGRD', currentline)
      if match:
        currentline_tval = currentline_list[7]
        p_val['UGRD'][str(int(current_press))]=currentline_tval

      match = re.search(':VGRD', currentline)
      if match:
        currentline_tval = currentline_list[7]
        p_val['VGRD'][str(int(current_press))]=currentline_tval

      sorted_pressvals=reversed(sorted(pressvals))  # Sort data levels in reverse order, that is,
highest to lowest.

#OUTPUT SECTION: output generated here although some output strings composed earlier in
program.

with open(output_file, "w") as fo:
  print('Writing to file: ', output_file)
  header_string='\n{0:9s}{1:25s}{2:12s}\n'.format(fcst_time,'forecast after model start at: ',
model_start)
  fo.write(header_string)
  header_string='{0:10s}{1:11s}{2:11s}{3:10s}\n\n'.format('Latitude: ', lat, 'Longitude: ', lon)
  fo.write(header_string)
  header_string='{0:9s}{1:9s}{2:10s}{3:8s}{4:8s}{5:8s}\n'.format(' P (hPa)', ' Hgt (m)', 'Tmp (K)', 'RH
(%)', 'U (m/s)', 'V (m/s)')
  fo.write(header_string)
  try:
    surface_string = '{0:7.1f}{1:9.1f} {2:7.2f} {3:7.1f} {4:7.2f}
{5:7.2f}\n'.format(float(surface_pressure), float(other_val['HGT'][current_surface]),
         float(other_val['TMP'][current_surface]), float(other_val['RH'][current_surface]),
         float(other_val['UGRD'][current_surface]), float(other_val['VGRD'][current_surface]))
    fo.write(surface_string)
  except KeyError:
    print("Surface data KeyError ", surface_pressure)
  try:
    tropo_string = '{0:7.1f}{1:9.1f} {2:7.2f} {3:7.1f} {4:7.2f} {5:7.2f}\n'.format(float(tropo_pressure),
float(other_val['HGT'][current_trop]),
            float(other_val['TMP'][current_trop]), float(other_val['RH'][current_trop]),
            float(other_val['UGRD'][current_trop]), float(other_val['VGRD'][current_trop]))
  except KeyError:
    print("troposphere data KeyError ", tropo_pressure)

  try:
    maxwind_string = '{0:7.1f}{1:9.1f} {2:7.2f} {3:7.1f} {4:7.2f}
{5:7.2f}\n'.format(float(maxwind_pressure), float(other_val['HGT'][current_maxwind]),
            float(other_val['TMP'][current_maxwind]), float(other_val['RH'][current_maxwind]),
            float(other_val['UGRD'][current_maxwind]), float(other_val['VGRD'][current_maxwind]))
  except KeyError:
    print("Maximum wind  data KeyError ", maxwind_pressure)

  last_pressure = 1999.0  # Initialize last_pressure with an improbable value as a start for the for
loop.

  for press_now in sorted_pressvals:
      if(press_now > surface_pressure):
        continue
      if((tropo_pressure > press_now and tropo_pressure < last_pressure) and (maxwind_pressure
> press_now and maxwind_pressure < last_pressure)):
         if(tropo_pressure > maxwind_pressure):
            fo.write(tropo_string)
            fo.write(maxwind_string)
         else:
            fo.write(maxwind_string)
            fo.write(tropo_string)
      elif(tropo_pressure > press_now and tropo_pressure < last_pressure):
         fo.write(tropo_string)
      elif(maxwind_pressure > press_now and maxwind_pressure < last_pressure):
         fo.write(maxwind_string)

      try:
        data_string = '{0:7.1f} {1:8.1f} {2:7.2f} {3:7.1f} {4:7.2f} {5:7.2f}\n'.format(int(press_now),
float(p_val['HGT'][str(int(press_now))]),
                   float(p_val['TMP'][str(int(press_now))]), float(p_val['RH'][str(int(press_now))]),
                    float(p_val['UGRD'][str(int(press_now))]), float(p_val['VGRD'][str(int(press_now))]))
        fo.write(data_string)
        last_pressure = press_now
      except KeyError:
        print("Key error for dictionary variable found at pressure level: ", str(int(press_now)))
```

List of Symbols, Abbreviations, and Acronyms

ARL	US Army Research Laboratory
DTC	Developmental Test Center
ETGB	Bergen, Germany
GFS	Global Forecast System
GRIB2	Gridded Binary, edition 2
METCM	computer meteorological message
NOAA	National Oceanic and Atmospheric Administration
NOMADS	National Operational Model Archive and Distribution System
URL	Uniform Resource Locator
UTC	coordinated universal time
WRF	Weather Research and Forecasting

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIR ARL IMAL HRA RECORDS MGMT RDRL DCL TECH LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	ARL RDRL CIE J COGAN